AMENDMENT

In the Claims

Please amend claims 1, 3, 4, 5, 10, 23-25, and 28-30 as follows.

1. (Currently Amended) A method for extending a hidden execution and storage mode of a processor in a computer system, comprising:

providing a mechanism to enable loading of a first event handler that is stored in an original set of firmware as supplied by an original equipment manufacture (OEM) of the computer system into a hidden memory space that is accessible to the hidden execution and storage mode but is not accessible to other operating modes of the processor;

providing a mechanism to enable a loading of ~~an~~ a second event handler that is not stored in ~~an~~ the original set of firmware as supplied by an original equipment manufacture (OEM) of the computer system into ~~a~~ the hidden memory space ~~that is accessible to the hidden execution and storage mode but is not accessible to other operating modes of the processor~~; and

enabling selective execution ~~executing~~ the first and second event ~~handler~~ handlers in response to an event that causes the processor to be switched to the hidden execution and storage mode to service the event.

2. (Original) The method of claim 1, wherein the hidden execution and storage mode comprises a System Management Mode (SMM) of a microprocessor.

3. (Currently Amended) The method of claim 1, wherein the mechanism for enabling loading and execution of the second event handler comprises:

providing an abstracted interface that enables a set of machine code corresponding to an event handler that is stored outside of any component(s) in which the original set of firmware is stored to be loaded into the hidden memory space;

redirecting an instruction pointer for the processor to execute the set of machine code to service the event while the processor is operating in the hidden execution and storage mode.

4. (Currently Amended) The method of claim 3, wherein the abstracted interface is published during a pre-boot process for the computer system to enable a driver to load the set of machine code corresponding to the second event handler prior to loading an operating system for the computer system.

5. (Currently Amended) The method of claim 1, wherein the mechanism for enabling loading of the second event handler comprises:

scanning for any firmware volumes that are materialized during a pre-boot process for the computer system to identify an existence of any firmware file containing an event handler that is compatible with the hidden execution and storage mode of the processor;

loading the event handler into the hidden memory space;

redirecting an instruction pointer for the processor to execute the event handler to service the event while the processor is operating in the hidden execution and storage mode.

6. (Original) The method of claim 1, further comprising:

loading an event handler management service into the hidden memory storage space;

registering one or more event handlers with the event handling management service;

loading said one or more event handlers into the hidden memory space;

redirecting an instruction pointer for the processor to begin execution of the event handler management service is response to the event that causes the processor to be switched to the hidden execution and storage mode; and

dispatching an event handler via the event handler management service to service the event.

7. (Original) The method of claim 6, wherein a plurality of event handlers are registered with the event handling management service and loaded into the hidden memory space, further comprising:

creating an ordered list of said plurality of event handlers;

dispatching a first event handler;

determining if the first event handler is an appropriate event handler for servicing the event and, if it is, executing the first event handler to completion to service the event; otherwise

dispatching a next event handler in the list and determining whether that event handler is an appropriate event handler and repeating this function until the appropriate event handler has been dispatched, whereupon that event handler is executed to completion to service the event.

8. (Previously Presented) The method of claim 7, wherein each of said plurality of event handlers comprise a set a machine code that is executed by the processor to service an error condition generated by a hardware component in the computer system that causes the event, and determining whether an event handler is the appropriate event handler for servicing the event comprises:

executing a first portion of the set of machine code corresponding to the event handler that was most recently dispatched that queries the hardware component corresponding to that event handler to determine if the error condition was caused by that hardware component; and

completing execution of the set of machine code for the event handler if it is determined that the error condition was cause by its corresponding hardware component, otherwise returning a value to the event handler management service indicating that the event handler is not the appropriate event handler to service the error condition.

9. (Original) The method of claim 6, further comprising authenticating the event handler before it is loaded into the hidden memory space.

10. (Currently Amended) The method of claim 6, wherein the original set of firmware includes one or more legacy event handlers including the first event handler, further comprising:

registering said one or more legacy event handlers with the event handling management service;

loading said one or more legacy event handlers into the hidden memory space accessible to the hidden execution and storage mode; and

dispatching at least one of said one or more legacy event handlers via the event handler management service to service the event.

11. (Original) The method of claim 6, wherein the computer system includes a plurality of processors, further comprising:

loading the service handler management service into a selected processor among said plurality of processors;

causing the selected processor to begin execution of the service handler management service in response to the event;

synchronizing all of said plurality of processors other than the selected processor and halting execution of a respective current operation for each of these other processors during execution of the service handler management service;

returning all of said plurality of processors to a previous processing mode to resume execution of their respective operations after the event has been serviced by an appropriate event handler.

12. (Original) The method of claim 1, further comprising enabling any legacy event handlers that are stored as machine code in the original set of firmware to be executed to service the event, as appropriate, in response to the event.

13. (Original) A method for extending a System Management Mode (SMM) of a microprocessor in a computer system, comprising:

publishing an interface during a pre-boot process for the computer system to enable a driver that is stored outside of components in which the original set of firmware is stored to provide a set of machine code comprising an event handler that is loaded into SMM memory (SMRAM) that is accessible to the microprocessor while in SMM;

switching the microprocessor to SMM in response to an SMM triggering event; and

executing the event handler to service the SMM triggering event.

14. (Original) The method of claim 13, further comprising:

loading an event handler management service into SMRAM;

registering one or more event handlers with the event handling management service;

loading said one or more event handlers into SMRAM;

redirecting an instruction pointer for the microprocessor to begin execution of the event handler management service is response to the SMM triggering event; and

dispatching an event handler via the event handler management service to service the event.

15. (Original) The method of claim 14, wherein a plurality of event handlers are registered with the event handling management service and loaded into SMRAM, further comprising:

creating an ordered list of said plurality of event handlers;

dispatching a first event handler;

determining if the first event handler is an appropriate event handler for servicing the SMM triggering event and, if it is, executing the first event handler to completion to service the event; otherwise

dispatching a next event handler in the list and determining whether that event handler is an appropriate event handler for servicing the SMM triggering event and repeating this function until the appropriate event handler has been dispatched, whereupon that event handler is executed to completion to service the SMM triggering event.

16. (Previously Presented) The method of claim 14, further comprising authenticating the event handler before it is loaded into SMRAM.

17. (Original) The method of claim 14, wherein the original set of firmware includes one or more legacy event handlers, further comprising:

registering said one or more legacy event handlers with the event handling management service;

loading said one or more legacy event handlers into SMRAM; and

dispatching at least one of said one or more legacy event handlers via the event handler management service to service a legacy event corresponding to the SMM triggering event.

18. (Original) The method of claim 13, further comprising:

scanning for any firmware volumes that are materialized during the pre-boot process for the computer system to identify an existence of any firmware file containing an event handler that is compatible with the SMM mode of the microprocessor;

loading the event handler into SMRAM; and

executing the event handler to service the SMM triggering event.

19. (Original) A method for handling a Platform Management Interrupt (PMI) event in a processor, comprising:

loading a PMI event-handling management service into memory accessible to the processor;

registering an entry point for the PMI event-handling management service;

enabling one or more PMI event handlers to be made accessible to the processor via the PMI event-handling management service; and

in response to the PMI event, vectoring the processor to begin executing the PMI event-handling management service at its entry point, wherein execution of the PMI event-handling management service performs the function of:

dispatching at least one of said one or more PMI event handlers to service the PMI event.

20. (Original) The method of claim 19, wherein the said one or more PMI event handlers are made accessible to the PMI event-handling management service by publishing a registration interface that enables registration of PMI event handlers with the PMI event-handling management service.

21. (Original) The method of claim 20, wherein a plurality of event handlers are registered with the PMI event handling management service, further comprising:

creating an ordered list of said plurality of event handlers;

dispatching a first event handler;

determining if the first event handler is an appropriate event handler for servicing the PMI event and, if it is, executing the first event handler to completion to service the event; otherwise

dispatching a next event handler in the list and determining whether that event handler is an appropriate event handler for servicing the PMI event and repeating this function until the appropriate event handler has been dispatched, whereupon that event handler is executed to completion to service the PMI event.

22. (Original) The method of claim 19, wherein the computer system includes a plurality of processors, further comprising:

loading the PMI event handler management service into a selected processor among said plurality of processors;

causing the selected processor to begin execution of the PMI event handler management service in response to the event;

synchronizing all of said plurality of processors other than the selected processor and halting execution of a respective current operation for each of these other processors during execution of the service handler management service;

returning all of said plurality of processors to a previous processing mode to resume execution of their respective operations after the PMI event has been serviced by an appropriate event handler.

23. (Currently Amended) A machine-readable medium having a plurality of machine instructions stored thereon that when executed by a processor in a computer system performs the operations of:

providing a mechanism to enable a loading of ~~an~~ a plurality of event ~~handler~~ handlers that ~~is~~ are not stored in an original set of firmware as supplied by an original equipment manufacture (OEM) of the computer system into a hidden memory space that is accessible to a hidden execution and storage mode of the processor but is not

accessible to other operating modes of the processor, the mechanism enabling event handlers from at least two different third party sources to be loaded into the hidden memory space; and

selectively executing ~~the~~ an appropriate event handler from among the plurality of event handlers in response to an event that causes the processor to be switched to the hidden execution and storage mode to service the event.

24. (Currently Amended) The machine-readable medium of claim 23, wherein the mechanism for enabling loading and execution of the plurality of event ~~handler~~ handlers comprises:

providing an abstracted interface that enables a set of machine code corresponding to an event handler that is stored outside of any component(s) in which the original set of firmware is stored to be loaded into the hidden memory space;

redirecting an instruction pointer for the processor to execute the set of machine code to service the event while the processor is operating in the hidden execution and storage mode.

25. (Currently Amended) The machine-readable medium of claim 23, wherein the mechanism for enabling loading and execution of the plurality of event ~~handler~~ handlers comprises:

scanning for any firmware volumes that are materialized during a pre-boot process for the computer system to identify an existence of any firmware file containing an event handler that is compatible with the hidden execution and storage mode of the processor;

loading the event handler into the hidden memory space;

redirecting an instruction pointer for the processor to execute the event handler to service the event while the processor is operating in the hidden execution and storage mode.

26. (Original) The machine-readable medium of claim 23, wherein execution of said plurality of machine instructions further performs the operations of:

loading an event handler management service into the hidden memory space;

registering one or more event handlers with the event handling management service;

loading said one or more event handlers into the hidden memory space;

redirecting an instruction pointer for the processor to begin execution of the event handler management service is response to the event that causes the processor to be switched to the hidden execution and storage mode; and

dispatching an event handler via the event handler management service to service the event.

27. (Original) The machine-readable medium of claim 26, wherein a plurality of event handlers are registered with the event handling management service and loaded into the hidden memory space, and execution of said plurality of machine instructions further performs the operations of:

creating an ordered list of said plurality of event handlers;

dispatching a first event handler;

determining if the first event handler is an appropriate event handler for servicing the event and, if it is, executing the first event handler to completion to service the event; otherwise

dispatching a next event handler in the list and determining whether that event handler is an appropriate event handler and repeating this function until the appropriate event handler has been dispatched, whereupon that event handler is executed to completion to service the event.

28. (Currently Amended) A computer system comprising:

a motherboard on which an original set of firmware is stored;

a memory operatively coupled to the motherboard in which a plurality of machine instructions are stored; and

a processor linked in communication with the memory, for executing the machine instructions to perform the operations of:

loading a first event handler that is stored in an original set of firmware as supplied by an original equipment manufacture (OEM) of the computer system into a hidden memory space that is accessible to the hidden execution and storage mode but is not accessible to other operating modes of the processor;

~~providing a mechanism to enable a~~ loading ~~of an~~ a second event handler that is not stored in ~~an~~ the original set of firmware as supplied by an original equipment manufacture (OEM) of the computer system into ~~a~~ the hidden memory space ~~that is accessible to the hidden execution and storage mode but is not accessible to other operating modes of the processor~~; and

enabling selective execution ~~executing~~ of the first and second event ~~handler~~ handlers in response to an event that causes the processor to be switched to the hidden execution and storage mode to service the event.

29. (Currently Amended) The computer system of claim 28, wherein ~~the mechanism for enabling~~ loading and execution of the second event handler comprises:

providing an abstracted interface that enables a set of machine code corresponding to an event handler that is stored outside of any component(s) in which the original set of firmware is stored to be loaded into the hidden memory space;

redirecting an instruction pointer for the processor to execute the set of machine code to service the event while the processor is operating in the hidden execution and storage mode.

30. (Currently Amended) The computer system of claim 28, wherein ~~the mechanism for enabling~~ loading and execution of the second event handler comprises:

scanning for any firmware volumes that are materialized during a pre-boot process for the computer system to identify an existence of any firmware file containing an event handler that is compatible with the hidden execution and storage mode of the processor;

loading the second event handler into the hidden memory space;

redirecting an instruction pointer for the processor to execute the second event handler to service the event while the processor is operating in the hidden execution and storage mode.